Onlineshop

Aufgaben

Ein Internet-Händler plant einen einfachen Onlineshop, über den eine Kundin oder ein Kunde (im Folgenden Kunde genannt), Produkte bestellen kann. Der Onlineshop soll als Client/Server-System mit einer Datenbank entwickelt werden.

1 Objektorientierte Entwicklung des Onlineshops
Der Softwareentwicklung liegen folgende Anforderungen zugrunde:
- Der Onlineshop lässt nur Bestellungen von registrierten Kunden zu, Bestellungen als Gast sind nicht möglich.
- Ein registrierter Kunde kann beliebig viele Produkte in seinen Warenkorb legen.
- Hat der Kunde die Produktauswahl beendet, kann er bestellen. Dabei werden die Positionen des Warenkorbs zu einem neuen Bestellobjekt hinzugefügt und im Warenkorb gelöscht.

Ein erstes UML-Klassendiagramm finden Sie in Material 1.

1.1 Beschreiben Sie das gegebene Klassendiagramm und erläutern Sie die Bedeutung der Notationselemente unter Beachtung der oben beschriebenen Anforderungen.

Hinweis: Die Stichworte Klasse, Objekt, Attribut, Methode, Beziehung, Multiplizität, Rolle sind sinnvoll zu verwenden.

(6 BE)

1.2 Die Kundin Erna Musterfrau mit der E-Mail-Adresse erna@musterfrau.de aus 65185 Wiesbaden möchte drei Packungen Druckerpapier zum Preis von je 3,95 Euro und zwei Patronen schwarze Druckertinte zum Preis von je 17,85 Euro kaufen und hat diese Produkte in ihren Warenkorb gelegt.
Zeichnen Sie ein UML-Objektdiagramm für dieses Szenario.

Hinweis: Fehlende Attributwerte sind sinnvoll zu ergänzen.

(5 BE)

1.3 Implementieren Sie die Klassen `Kunde` und `Warenkorb` mit den Konstruktoren und öffentlichen Methoden, jedoch ohne private Methoden.

Hinweise: Die Klasse `Kundendaten` beinhaltet alle nötigen Daten, um ein neues Kundenobjekt zu initialisieren. Der Konstruktor der Klasse `Kunde` generiert eine neue Kundennummer, initialisiert die Kundendaten (private Methode `initKundendaten()`) und generiert ein Passwort (mittels der Methode `generierePw()`). Die Methode `hinzufuegenPosition()` der Klasse `Warenkorb` prüft, ob es bereits eine Position für das übergebene Produkt gibt. Ist bereits eine Position vorhanden, wird die Menge zur bereits existierenden Menge addiert. Gibt es noch keine Position für das Produkt, wird eine neue angelegt. Die Dokumentationen der Klassen `List` und `Kundendaten` sind in Material 2 zu finden.

(9 BE)

1.4 Möchte ein Kunde die ausgewählten Produkte im Warenkorb bestellen, so wird die Methode `bestellen()` der Klasse `Shop` aufgerufen. Die Methode realisiert folgenden Ablauf:

- Zunächst wird die Verfügbarkeit der ausgewählten Produkte anhand der Lagermengen geprüft.
- Ist eines der Produkte nicht in der gewünschten Menge verfügbar, wird keine Bestellung angelegt und die Methode gibt `false` zurück.
- Sind alle Produkte verfügbar, so wird eine neue Bestellung angelegt und jede Position des Warenkorbs wird in die Bestellung übertragen sowie jeweils im Warenkorb gelöscht. Danach werden die Lagermengen anhand dieser Bestellung aktualisiert und die Methode gibt `true` zurück.

Entwickeln und zeichnen Sie ein UML-Sequenzdiagramm für die Methode `bestellen()` in das Material 3.

Hinweise: Die Prüfung, ob ein Warenkorb überhaupt Produkte enthält, findet vor dem Aufruf der Methode `bestellen()` statt. Die Methode `updateLagermengen()` der Klasse `Shop` aktualisiert die Lagermengen aller in der Bestellung enthaltenen Produkte.

(10 BE)

1.5 Um eine Bestellung zu tätigen, soll zukünftig zwingend eine Zahlungsart für jede Bestellung gespeichert werden. Dazu wurden folgende neue Anforderungen erfasst:

- Der Shop kennt mehrere Zahlungsarten.
- Jede Zahlungsart besitzt eine eindeutige ID und eine Bezeichnung.
- Die Zahlungsart Vorkasse besitzt als Attribut die IBAN des Shops sowie die Zahlungsfrist (Anzahl Tage, in der die Vorauszahlung geleistet werden muss).
- Die Zahlungsart PayPal beinhaltet die Webadresse, zu der weitergeleitet wird, um die PayPal-Zahlung vorzunehmen, als öffentliche Klassenkonstante.
- Bei Zahlungsart Kreditkarte soll der Anbieter als String gespeichert werden.
- Die Zahlungsart soll sofort beim Anlegen der Bestellung gespeichert werden.

Modellieren Sie ein erweitertes UML-Klassendiagramm und zeichnen Sie die Änderungen vorhandener Klassen sowie neue Klassen mit Attributen (Sichtbarkeit und Datentypen), Methoden (Sichtbarkeit, Parameter und Rückgabetypen) und den Assoziationen (Navigierbarkeit, Multiplizitäten und Rollen).

Hinweise: Get- und set-Methoden müssen nicht dargestellt werden. Vererbung ist sinnvoll einzusetzen.

(6 BE)

1.6 Die Methode `berechnePruefziffer()` der Klasse `Shop` berechnet über einen Algorithmus die Prüfziffer zur übergebenen Kartennummer. Eine Beispielrechnung finden Sie in Material 4. Die Vorgehensweise zur Berechnung ist wie folgt:

- Von der ersten Ziffer der Kreditkartennummer wird diese nach rechts bis zur vorletzten Stelle durchlaufen, wobei der Wert der ersten und von dort ausgehend jeder zweiten Ziffer verdoppelt wird.
- Von den Produkten (Ziffer mal Faktor) wird jeweils die Quersumme ermittelt und aufaddiert.
- Von dieser Summe wird der Modulo 10-Wert berechnet.
- Im letzten Schritt wird das Ergebnis des vorherigen Schrittes von 10 subtrahiert. Das Ergebnis ist die Prüfziffer.

Entwickeln und zeichnen Sie ein Struktogramm für diesen Algorithmus.

(8 BE)

1.7 Kunden müssen sich mit einem Shop-Client über das Internet mit dem Shop-Server verbinden, um online Produkte zu bestellen. Sie finden ein erweitertes UML-Klassendiagramm der Serverseite in Material 5.

1.7.1 Erklären Sie den Begriff Thread und beschreiben Sie die Arbeitsweise der Klasse `Server` in Material 5.

(4 BE)

1.7.2 Das Klassendiagramm aus Material 5 soll um neue Anforderungen erweitert werden:

- Wenn eine Wartung des Shopsystems erforderlich ist, dürfen keine neuen Threads mehr gestartet werden. Clients sollen in diesem Fall die Nachricht „Shop nicht verfügbar" erhalten. In der Klasse `Server` soll es eine private Methode `wartung()` geben, die eine Systemvariable überprüft, die im Wartungsfall gesetzt werden kann.
- Die Wartung kann nur erfolgen, wenn kein Kunde mehr im Onlineshop aktiv ist. Eine Methode der Klasse `Server` soll zu diesem Zweck die Anzahl der aktiven Threads ausgeben.
- Bevor ein Thread ordnungsgemäß terminiert, muss er sich beim Server abmelden.
- Damit die Anzahl der aktiven Threads angezeigt werden kann, gibt es einen `AnzeigeThread`, der im Konstruktor des Servers gestartet wird und die Anzahl der aktiven Threads auf der Serverkonsole anzeigt.

Entwickeln Sie ein erweitertes UML-Klassendiagramm für diese Anforderungen.

Hinweis: Das UML-Klassendiagramm in Material 5 kann als Vorlage genutzt werden.

(4 BE)

1.7.3 Implementieren Sie die Klasse `Server` auf Basis Ihres Modells aus Aufgabe 1.7.2.

Hinweis: Die private Methode `wartung()` ist nicht zu implementieren.

(6 BE)

1.7.4 Es ist damit zu rechnen, dass viele Kunden zeitgleich über den Onlineshop bestellen werden. Erläutern Sie anhand des Ablaufs der Methode `bestellen()` aus Aufgabe 1.4 typische Probleme, die im Mehrbenutzerbetrieb auftreten können.

(2 BE)

2 Entwicklung einer Datenbank für den Onlineshop
Die Daten des Onlineshops sollen in einer relationalen Datenbank gespeichert werden. Ein Entity-Relationship-Modell (ERM) finden Sie in Material 6.

2.1 Nennen und erläutern Sie vier Vorteile, die der Einsatz eines Datenbankmanagementsystems im Rahmen eines Softwareprojekts bietet.

(4 BE)

2.2 Überführen Sie das ERM (Material 6) in das relationale Modell in der 3. Normalform.

Hinweis: Alle Relationen sind in der Schreibweise `Relation(PK, Attribut, ..., FK#)` anzugeben.

(7 BE)

2.3 Die Daten der Datenbank sollen ergänzt, aktualisiert bzw. ausgewertet werden.

Hinweis: Die Funktion `NOW()` liefert das aktuelle Tagesdatum, die Funktionen `YEAR()` bzw. `MONTH()` liefern zum übergebenen Datum das jeweilige Jahr bzw. den Monat.

2.3.1 Geben Sie eine SQL-Anweisung an, die den Listenpreis des Produkts mit der Produktnummer 4713 um 5 % erhöht.

(2 BE)

2.3.2 Für den Kunden Benjamin Wagner, dessen Name nur einmal in der Datenbank vorhanden ist, soll eine Bestellung mit der Bestellnummer 4348 und dem heutigen Datum angelegt werden. Formulieren Sie die dafür erforderliche SQL-Anweisung.

(3 BE)

2.3.3 Entwickeln Sie eine SQL-Anweisung, die das umsatzstärkste Produkt des aktuellen Monats mit Produktnummer, Bezeichnung und seinem Umsatz ermittelt.

(5 BE)

2.3.4 Der Onlineshop möchte Produktempfehlungen geben.
Entwickeln Sie eine SQL-Anweisung, die alle Produkte mit Produktnummer und Bezeichnung ermittelt, die zusammen mit dem Produkt mit der Produktnummer 4349 im letzten Jahr bestellt wurden.

(4 BE)

2.3.5 Produkte mit einer Lagermenge unter 20 sollen nachbestellt werden.
Entwickeln Sie eine SQL-Anweisung, die für jedes dieser Produkte den günstigsten Lieferanten (Ausgabe des Firmennamens) ermittelt.

(5 BE)

2.4 Das ERM aus Material 6 soll um die Speicherung der Daten von Rücksendungen ergänzt werden. Eine Rücksendung bedeutet, dass das Produkt nicht gekauft wird.
Um eine Rücksendung aufzugeben, muss der Kunde einen Rücksendeschein drucken, ausfüllen und der Sendung beilegen. Sie finden ein Beispiel für einen Rücksendeschein in Material 7.
Viele Kunden füllen den Rücksendeschein sehr oberflächlich aus, d. h. es gibt Rücksendungen ohne Angabe einer Rechnungsnummer oder Informationen zu den Produkten.
Die Annahmestelle der Sendung übermittelt dem Onlineshop folgende Daten:
- ID der Annahmestelle
- Bezeichnung der Annahmestelle
- Annahmedatum

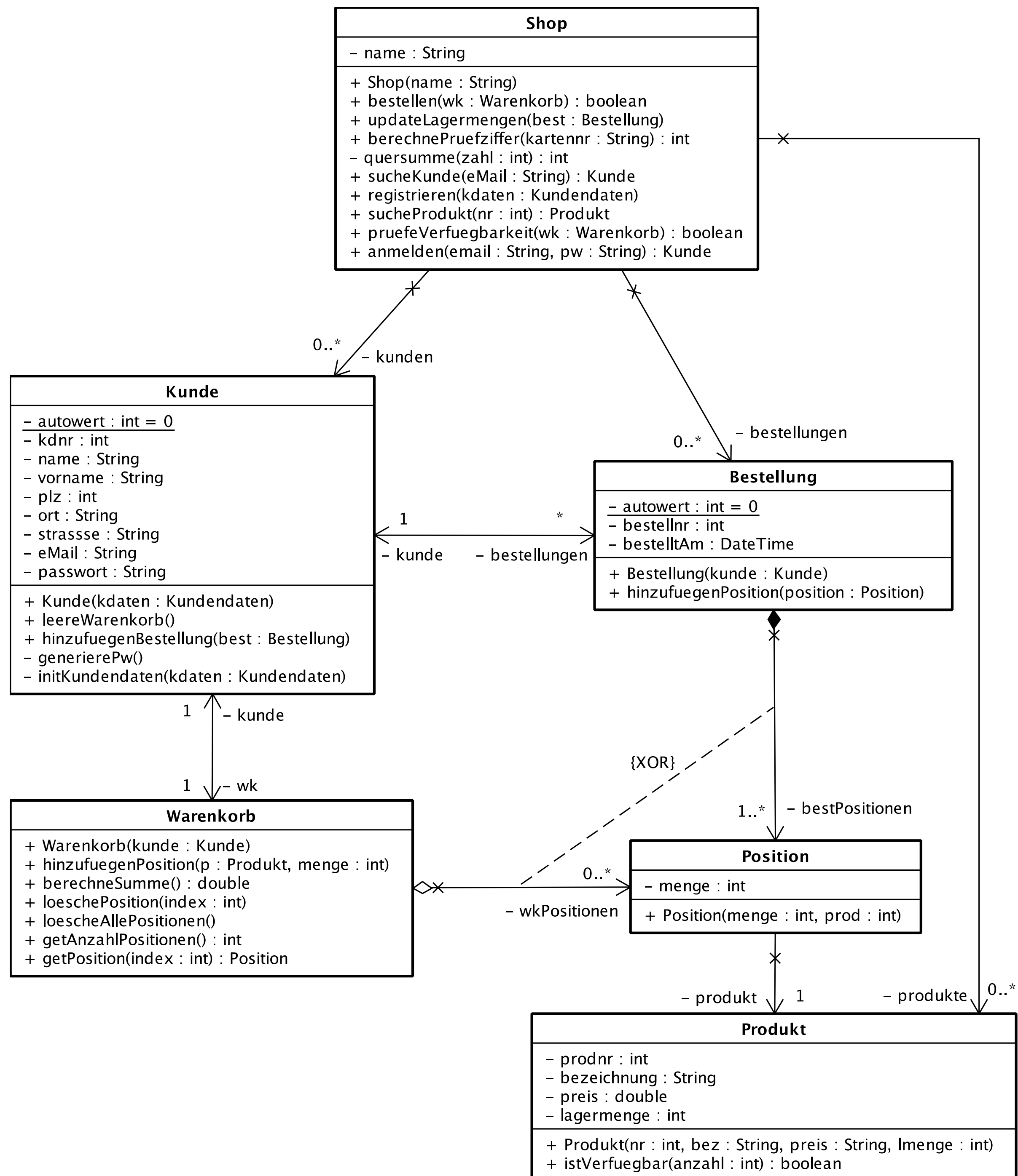
Bei Eingang der Rücksendung wird das Eingangsdatum beim Onlineshop gespeichert.
Modellieren und zeichnen Sie unter Berücksichtigung des gegebenen Modells (Material 6) ein ER-Diagramm, das die genannten Anforderungen umsetzt.

Hinweise: Die Angaben Ort, Datum sowie die Unterschrift werden nicht in der Datenbank gespeichert, sondern mit dem Beleg archiviert. Das Diagramm ist mit Entitätstypen, Attributen und Beziehungen mit Kardinalitäten in der [min, max]-Form darzustellen. Bei der Erstellung eines neuen Diagramms sind nur die Erweiterungen und Änderungen zum gegebenen ER-Modell darzustellen.

(10 BE)

Material 1

UML-Klassendiagramm



Hinweis: Die XOR-Einschränkung sagt aus, dass ein Positionsobjekt zu einem Zeitpunkt entweder Teil eines Warenkorb- oder Teil eines Bestellungsobjekts sein kann, aber niemals Teil von beiden.

Material 2

Klassendokumentationen

Klasse `Kundendaten`

Bei der Klasse `Kundendaten` handelt es sich um eine Hilfsklasse, um in kompakter Form alle Kundendaten übergeben zu können. Die Bezeichner der Attribute sind selbsterklärend. Auf jedes Attribut kann mittels get- und set-Methode zugegriffen werden.

Kundendaten
- name : String - vorname : String - strasse : String - plz : int - ort : String - email : String

Klasse `List`

`List<T>()`
erzeugt eine generische Liste mit Elementen des Typs `T`.

`add(obj: T)`
hängt das Objekt `obj` vom Typ `T` am Ende der Liste an.

`add(index: int, obj: T)`
fügt das Objekt `obj` vom Typ `T` an der Position `index` in die Liste ein.

`clear()`
löscht alle Elemente der Liste.

`contains(obj: T): boolean`
liefert `true`, wenn das Objekt `obj` in der Liste enthalten ist, sonst `false`.

`get(index: int): T`
liefert das Listenelement an der Position `index` zurück bzw. `null`, falls `index` negativ oder größer gleich der Anzahl der momentan enthaltenen Elemente ist.

`remove(index: int): T`
entfernt das Listenelement an der Position `index`. Liefert das entfernte Element zurück bzw. `null`, falls `index` negativ oder größer gleich der Anzahl der momentan enthaltenen Elemente ist.

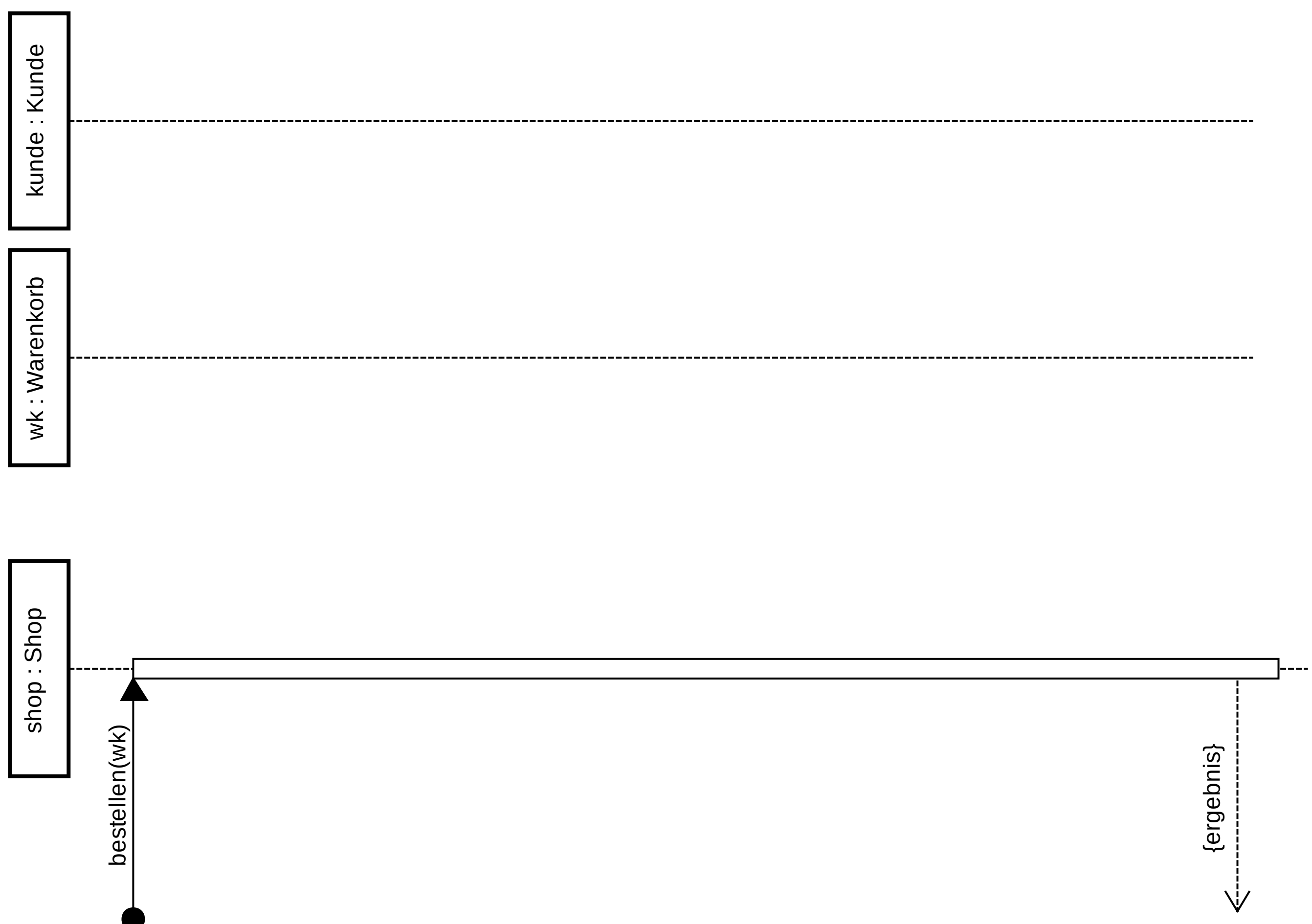
`remove(obj: T): boolean`
entfernt das Objekt `obj` aus der Liste. Falls `obj` mehrmals in der Liste enthalten ist, wird nur das erste Vorkommen entfernt. Der Rückgabewert ist `true`, falls das Objekt gefunden und entfernt wurde, sonst `false`.

`size(): int`
liefert die Anzahl der Elemente in der Liste zurück.

List<T>
+ List<T>() + add(obj : T) + add(index : int, obj : T) + clear() + contains(obj : T) : boolean + get(index : int) : T + remove(index : int) : T + remove(obj : T) : boolean + size() : int

Material 3

Vorlage UML-Sequenzdiagramm



Material 4

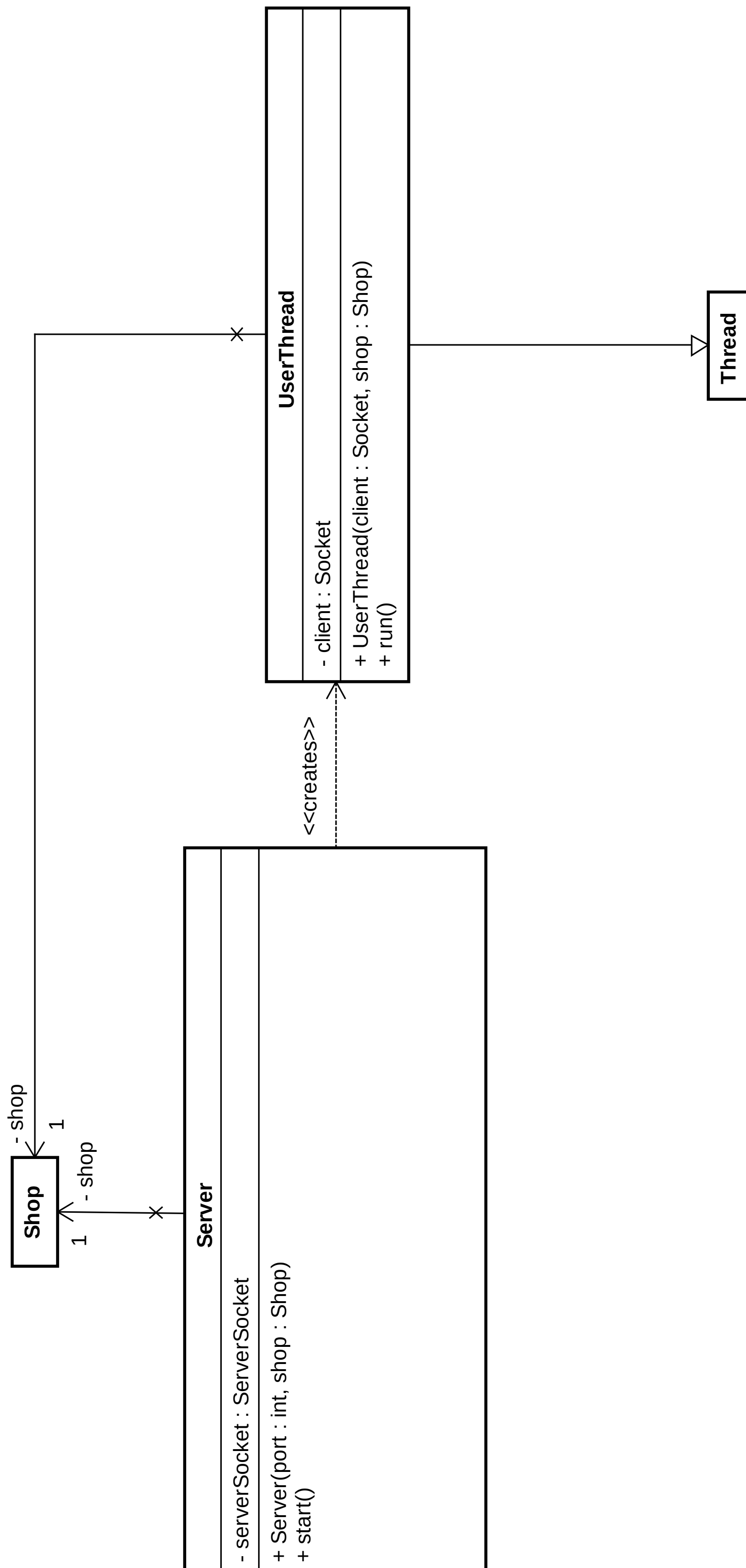
Beispiel zur Berechnung der Prüfziffer einer Kreditkartennummer

Die Kreditkartennummer 4683 4578 2937 6514 enthält an der letzten Stelle die Prüfziffer.

Ziffer	Faktor	Produkt	Quersumme
4	2	8	8
6	1	6	6
8	2	16	7
3	1	3	3
4	2	8	8
5	1	5	5
7	2	14	5
8	1	8	8
2	2	4	4
9	1	9	9
3	2	6	6
7	1	7	7
6	2	12	3
5	1	5	5
1	2	2	2
Summe der Quersummen			86
Modulo 10			6
Differenz (= Prüfziffer)			10-6 = 4

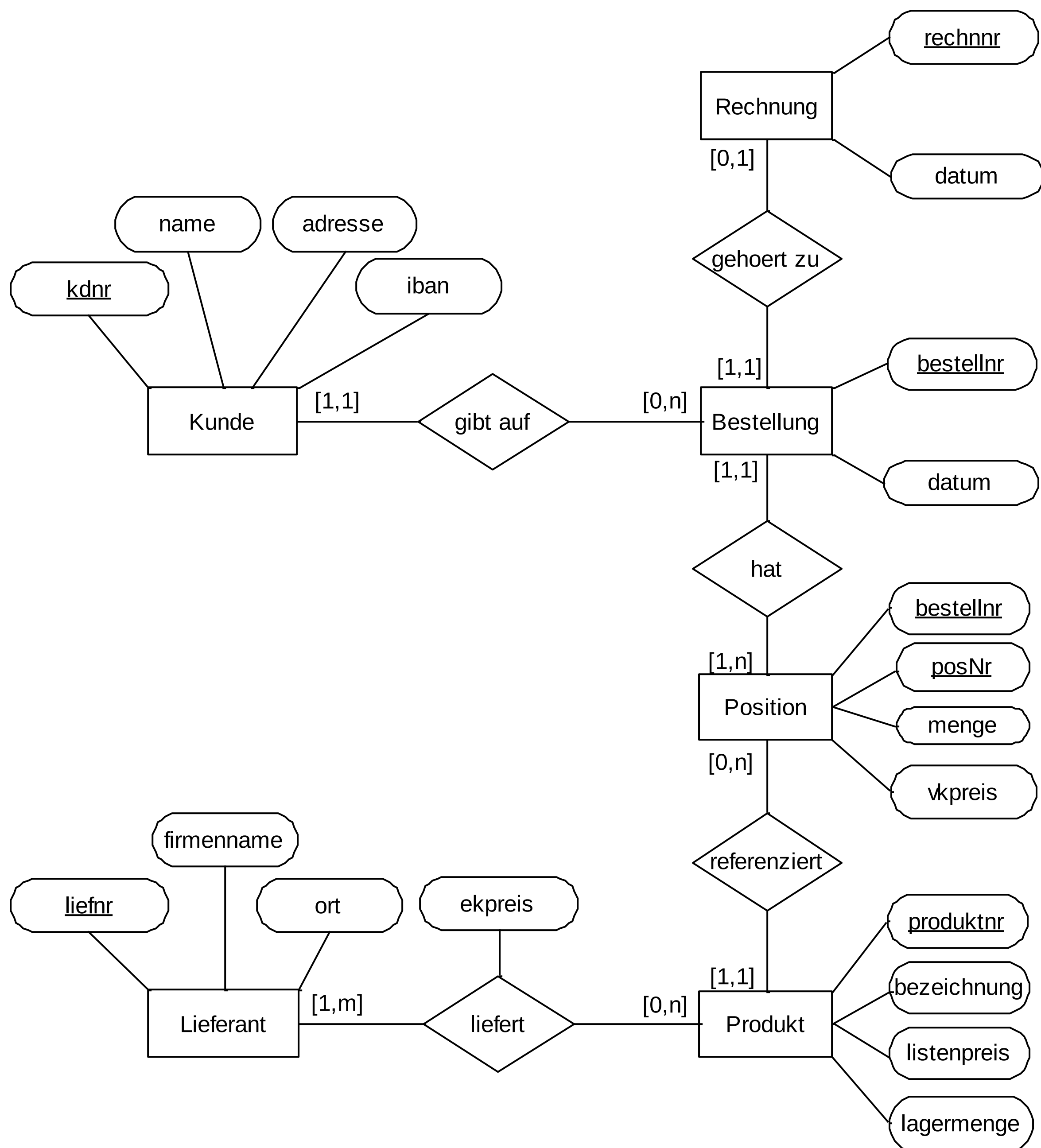
Material 5

UML-Klassendiagramm der Serverseite



Material 6

Entity-Relationship-Modell



Material 7

Beispiel eines Rücksendescheins

Rücksendeschein

Ultimate Shopping

Identnummer 84.339.083.8927

Absender
Kundennummer 7296
Erna Musterfrau
Kirschgasse 87
65185 Wiesbaden

Kundenwunsch Rückzahlung (bitte ankreuzen)

- ☐ Bitte überweisen Sie auf mein Bankkonto
- ☐ Ich erbitte eine Gutschrift

Rücksendung folgender Produkte

	Menge	**Produkt-nummer**	**Produktbezeichnung**	**Rechnungs-nummer**	**Rücksendegrund**
1					
2					
3					
4					

____________________________ ____________________________

Ort, Datum Unterschrift

Hier folgen der Einlieferungsschein und der Paketaufkleber. Sie wurden aus Vereinfachungsgründen weggelassen. Je nach gewählter Versandoption (DHL, Hermes oder DPD) sieht dieser Abschnitt anders aus.